FFP Version 2,0:
An Implementation of COSMIC Functional Size Measurement Concepts

By: Alain Abran

COSMIC Co-manager[1]

Keynote presentation at the FESMA 99 Conference
Amsterdam, Oct. 7, 1999

Abstract

The Full Function Points functional size measurement method was first released in the fall of 1997. Since this initial presentation, significant improvements to the description of functional size measurement concepts have been achieved by the COSMIC group. In its second release, the Full Function Points method has been enhanced significantly in order to implement the findings of the COSMIC group. Highlights of the improvements will be presented, including the clarifications to the measurement process model as well as enhancements to the functional size model and to the measurement procedures.

1. INTRODUCTION

Software functional size measurement is regarded as a key aspect in the production, calibration and use of software engineering productivity models because of its independence from technologies and implementation decisions. In 1997 Full Function Points (FFP) was proposed as a method for measuring the functional size of real-time and embedded software. Since its introduction, the FFP measurement method has been field tested, and used, in many organizations providing further feedback on its applicability in software fields outside of its initial scope.

The Full Function Points measurement method has not only been applied to real-time or embedded software but also to a variety of technical and system software and to some MIS software. Applying the method to such a wide range of software brought forward:

a) The need to improve the mechanism allowing well defined mappings to be established between the standard artifacts produced while using a specific software engineering methodology and the Base Functional Components (BFC's) [ISO/IEC 14143-1] used to measure the functional size of the corresponding software.

b) The need to refine the concept of "software boundary" in order to address functional users requirements allocated not only to the pieces of software interacting with end users but also to pieces of supporting software which are part of the operating environment; all pieces being part of a given project.

c) The need to simplify the set of BFC's used to measure the functional size of software.

[1] (COSMIC team: Alain Abran, Charles Symons – co-manager, Carol Dekkers, Jean-Marc Desharnais, Peter Fagg, Paul Goodman, Pam Morris, Serge Oligny, Jolijn Onvlee, Risto Nevalainen, Grant Rule, Denis St Pierre)

In parallel, ISO has issued a new standard (ISO/IEC 14143-1) on the definition and concepts of functional size measurement methods, while a group of international experts, COSMIC (COmmon Software Measurement International Consortium), has established the design criteria for the next generation of software size measurement methods. In particular, significant improvements to the description of functional size measurement concepts have been achieved by the COSMIC group.

These ISO and COSMIC requirements were used as design specifications to produce Version 2,0 of the Full Function Points measurement method, which was pre-released in September 1999 for the purpose of technology transitioning to industrial partners and for fine-tuning in industrial contexts.

This presentation describes some of the key improvements introduced in this first implementation of the COSMIC principles through new features of FFP version 2,0 including: a generic software model adapted to the purpose of functional size measurement, a two-phase approach to functional size measurement (mapping and measurement), and a simplified set of base functional components (BFC). Through its generic software model, version 2,0 of the COSMIC - FFP measurement method is applicable to a broad range of software including embedded, MIS, middleware and system software.

2. COSMIC – FFP

FFP version 1,0 has been used in multiple types of software ranging from MIS to telecommunications and to defense related software. This diversity of contexts, and associated vocabulary specific to each software community, has imposed much rigor to ensure that underlying measurement concepts be stated in an explicit manner independent of a specific domain of application and to ensure that the transferability of the concept to be measured would be grasped similarly across multiple communities of measurers.

Therefore, many of the improvements introduced in COSMIC – FFP Measurement manual are marked by explicit measurement related concepts. Such an approach means that the concepts and general principles of the software attribute to be measured, that is size of the functional user requirements, are distinguished from the rules and procedures used to implement them. The measurement method therefore gains in flexibility and usability by allowing the practitioners to grasp quickly the aim of the rules within the context of a specified instantiation of a measurement principle and, consequently, adapt them to the measurement context of their organization. This has lead in particular to a major change in the structure of the documentation of the measurement method, moving away from the first generation of a ‘counting manual’, to the next generation of a ‘measurement standards manual’, as could be expected in an engineering discipline, for example.

2.1 A measurement process model

In essence, the COSMIC - FFP measurement method consists in the application of measurement principles through a set of rules and procedures to a given piece of software; the result of the application of these rules and procedures is a numerical figure representing the functional size of the software. The method is designed to be independent of the implementation decisions embedded in the operational artifacts of the software to be measured. To achieve this characteristic, the method is applied to a generic software model onto which actual artifacts of the software to be measured are mapped. Figure 2.1 depicts this process.

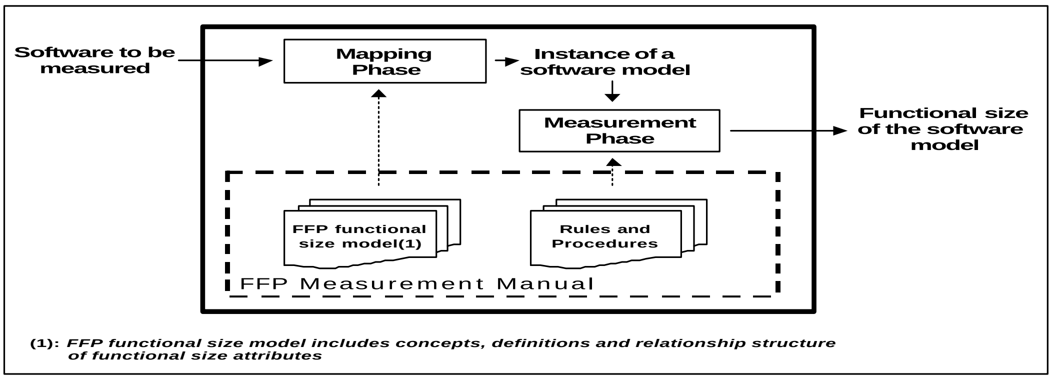

Software to be measured
Mapping Phase
Instance of a software model
Measurement Phase
Functional size of the software model
FFP functional size model(1)
Rules and Procedures
FFP Measurement Manual
(1): FFP functional size model includes concepts, definitions and relationship structure of functional size attributes

Figure 2.1 – FFP Measurement process model

The FFP measurement rules and procedures are then applied to this instantiated FFP model in order to produce a numerical figure representing the functional size of the software.

Therefore, two distinct and related phases are necessary to measure the functional size of software: <u>mapping</u> the artifacts of the software to be measured onto an FFP software model and then <u>measuring</u> the specific elements of this software model.

The mapping phase takes as input the artifacts of the software to be measured (as they are found/documented within the organization) and produces as output an instance of an FFP software model.

This model illustrates that, prior to applying the measurement rules and procedures, the software to be measured must be mapped onto a specific FFP software model that captures the concepts, definitions and relationships (functional structure) required for a functional size measurement exercise.

2.2 A generic software model

A key aspect of software functional size measurement lies in the establishment of what is considered to be part of the software and what is considered to be part of the operating environment of the software. As a functional size measurement method, COSMIC - FFP aims at measuring the size of software based on identifiable functional requirements. Depending on how these requirements are allocated, the resulting software might be implemented in a number of pieces. While all the pieces exchange data, they will not necessarily operate at the same “level”. The COSMIC - FFP software context model, illustrated in Figure 2.2, recognizes this general configuration by providing rules to identify different LAYERS of software.

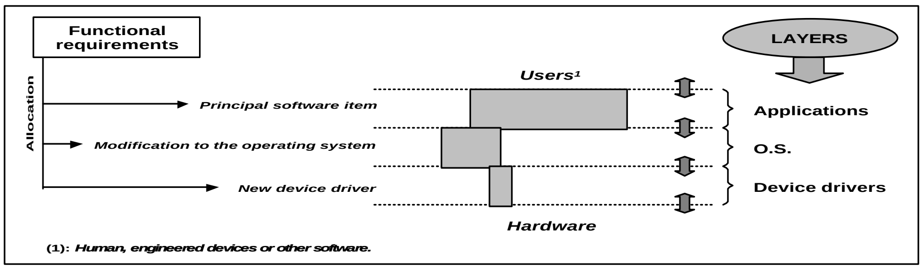


Figure 2.2 – COSMIC Software context model

As illustrated in Figure 2.2, the functional requirements in this example are allocated to three distinct pieces, each exchanging data with another through a hierarchical organization. One piece of the software lies at the application level and exchanges data with the software's users and with a second piece lying at the operating system level. In turn, this second piece of the software exchanges data with a third piece lying at the device driver level. This last piece then exchanges data directly with the hardware. The Full Function Points context model associates each level with a specific LAYER. Each layer possesses an intrinsic boundary for which specific users are identified. The functional size of the software described through the functional requirements is, therefore, the sum of the functional sizes of the pieces where those requirements have been allocated.

Four important concepts relating to this model are defined.

Concept 1 – Layers
The software to be measured can be partitioned into one or more pieces so that each piece operates at a different level of functional abstraction in the software's operating environment and there is a hierarchical relationship between each particular level of abstraction based on the data exchanged between them. Each such level is designated as a distinct layer. Each layer encapsulates functionality useful to the layer lying above itself in the hierarchical relationship and uses the functionality provided by the layer lying under itself in this relationship. A further example is provided in Annex A.

Concept 2 – Boundary
Within each identified layer, the piece of software to be measured can be clearly distinguished from its surrounding peers by a boundary. Furthermore, an implicit boundary exists between each identified layer. The software boundary is therefore a set of criteria, perceived through the functional requirements of the software, which allows a clear distinction to be made between the items that are part of the software (inside the boundary) and the items that are part of the software operating environment (outside the boundary). By convention, all users of a piece of software lie outside the boundary of this software. A example is provided in Annex B.

Concept 3 – Software users
Within each identified layer, it is possible to identify one or more users benefiting from the functionality provided by the piece of software lying inside the layer. By definition, users can be human beings, <u>engineered devices or other software</u>. Also by definition, pieces of the measured software lying inside the immediate neighboring layers are considered as users (considered as "other software"). Refer to Annex B.

Concept 4 - Functional Requirements
Software purposes can be formally described through a finite set of requirements. The parts of these requirements describing the nature of the functions to be provided are designated as functional requirements and are used as the exclusive perspective from which the functional size of the software is to be measured. The parts of the requirements describing how the software functions are to be implemented are NOT considered for the purposes of measuring the functional size of the software.

Figure 2.3 illustrates the software model proposed by the COSMIC - FFP measurement method. This model describes the perspective from which the pieces of software identified within each layer are perceived for the purpose of functional size measurement.

According to this model, software functional requirements are implemented by a set of functional processes. Each of these functional processes is an ordered set of sub-processes performing either a data movement or a data transform.

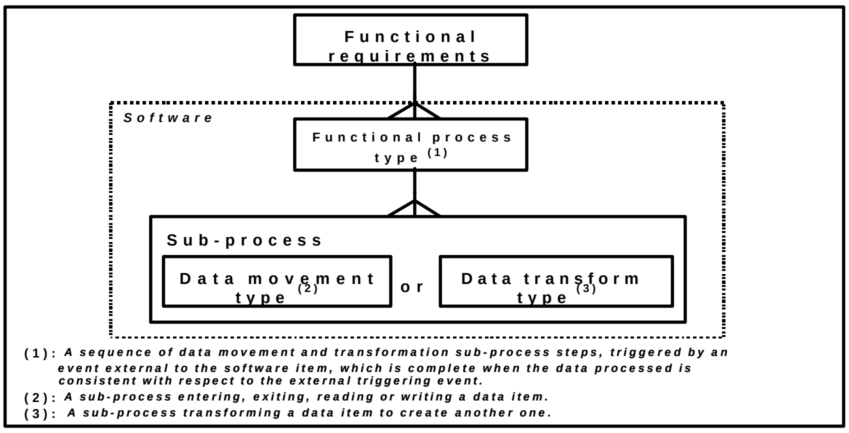

Functional requirements
Software
Functional process type (1)
Sub-process
Data movement type (2)
or
Data transform type (3)
(1): A sequence of data movement and transformation sub-process steps, triggered by an event external to the software item, which is complete when the data processed is consistent with respect to the external triggering event.
(2): A sub-process entering, exiting, reading or writing a data item.
(3): A sub-process transforming a data item to create another one.

Figure 2.3 – COSMIC Generic software model

The COSMIC - FFP generic software model distinguishes four types of data movement sub-process: entry, exit, read and write. All data movement sub-processes move only one piece of data. *Entries* move the piece from outside the software boundary to the inside; *Exits* move it from inside the software boundary to the outside; reads and writes move it without crossing the software boundary. These relationships are illustrated in Figure 2.4. Another representation is presented in Annex C. It must be observed that the level of granularity of the measurement of the data movements is at the data types rather than at the data instances, which means that the data movements from multiple types of sensors would be taken into consideration in the measurement model, and not the multiple sensors of the same types.

By using the concepts, definitions and structure of the COSMIC - FFP measurement method, the artifacts of a piece of software are mapped onto the FFP software model, thereby instantiating it. This instantiated model will contain all the elements required for measuring its functional size, while hiding information not relevant to functional size measurement.

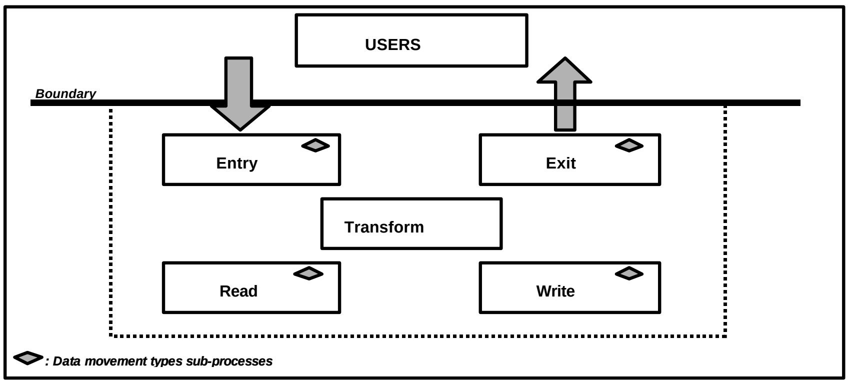


Figure 2.4 – COSMIC - FFP sub-process types

2.3 Base Functional Components and Standard Unit (Etalon)

The COSMIC - FFP measurement phase takes as input an instance of a software model and, using a defined set of rules and procedures, produces a numerical figure the magnitude of which is directly proportional to the functional size of the model, based on the following principle:

The functional size of a piece of software is directly proportional to the number of its data-moving sub-processes.

FFP Measurement principle

By convention, this numerical figure is then extended to represent the functional size of the software itself.

Two elements characterize this set of rules and procedures: the BFC's which are the arguments of the measurement functions and the standard unit (*etalon)*, which is the yardstick defining one unit of size (one FFP).

Base Functional Components
COSMIC - FFP uses only four base functional components: entry, exit, read and write. Data transform sub-processes are not used as a base functional component, it is assumed that the functionality of this type of sub-process is represented among the other four type of sub-processes.

Standard Unit (Étalon[2])
The standard unit, that is, 1 Full Function Point, is defined by convention as one single data movement at the sub-process level.

It is worth noting that there is no upper limit to the functional size of a piece of software and, notably, there is no upper limit to the functional size of any of its measured functional processes.

For the purpose of transitioning the method to industry environments, the number of data attributes manipulated by each sub-process will be registered. This will allow further analysis to be conducted concerning the size to be assigned: a) to each sub-processes, and b) to each class of sub-processes. Such an analysis will consider the benefits and weaknesses of two measurement approaches: direct measurement where each identified sub-process is assigned a pre-determined and invariant size, and indirect measurement where each identified sub-process is assigned a size based on some sort of weighting function(s).

2.4 Scalability

From the level of granularity offered by the sub-processes types, the COSMIC - FFP measurement method offers a fully scalable result through the use of its aggregation function.

Thus, the functional size of any functional process is defined as the arithmetic sum of the size of its constituent sub-processes. Results can also be aggregated by layers and, by extension, to the whole of the inputted software model by simply adding the functional size of the constituent elements.

[2] In French, an Étalon is the reference of a standard unit of measurement, eg. the 'étalon' for the standard unit of 1 meter, the 'étalon' for the standard unit of time - 1 second.

2.5 Simplification of the measurement model

The first version of FFP published in 1997 had kept the initial model of the IFPUG method with its five (5) function types, to which the four function types described as ‘Entry’, ‘Exit’, ‘Read’ and ‘Write’ had been added to handle in a proper way the control functions types typical of real-time and embedded software; this meant a total of 9 function types in version 1,0. Since then, industry feedback was almost unanimous: the function types added in version 1,0 were of a much more generic nature than initially expected from their design, which meant that they could also be used to take into account the functionality of the management functions of most types of software. The need to simplify the set of BFC’s used to measure the functional size of software. The COSMIC – FFP model has therefore been significantly simplified in keeping in its measurement only the four function types (or data movement types) as described previously, thereby eliminating the need for mastering of the IFPUG rules, and the full independence of COSMIC - FFP in terms of evolution and standardization.

3. CONCLUDING REMARKS

Functional size measurement of software emerged 20 years ago from the empirical results gathered on a sample of MIS applications [*Albrecht 1979*]. As it gained wider acceptance by practitioners in the 80’s, the methods then available have been regularly criticized, notably for their inability to correctly measure the size of real-time software. Although many alternatives have been proposed from the mid-’80 to the mid-’90 to address this problem, none of them seemed to have gained sufficient recognition from the practitioners to be used on a regular basis across a large number of organizations in many countries.

Full Function Points were initially proposed in 1997 as a public domain alternative to solve this persistent difficulty. Since then, field tests and repeated usage in many organizations have demonstrated that FFP offers meaningful results not only to measure the functional size of real-time and embedded software but also to measure the functional size of a wide range of technical and system software and, in some cases, of MIS software as well.

In its second version, Full Function Points has been enhanced significantly in order to implement the findings of the COSMIC group. Highlights of the improvements were presented, including the implementation of the most innovative contributions of COSMIC. In addition, improvements have been made to the structure of the traditional ‘Counting Practices Manual’ in order to transform it into a manual of Measurement standards, as could be expected to be structured in an engineering discipline.

Technology transitioning is currently being set-up with various partners to ensure its full transferability into a large variety of industrial contexts and its proper fine tuning.

The COSMIC - FFP measurement method does not presume to measure all aspects of software size. Thus, other dimensions of software “size” are not captured by this measurement method, such as technical requirements. A constructive debate on this matter would first require commonly agreed upon definitions of the other elements within the ambiguous concept of “size” as it applies to software. Such definitions are still, at this point, the object of further research and much debate. It is part of the aims of the COSMIC group to tackle these important issues and eventually come up with additional types of measurement methods to capture these other perspectives of software size (in addition to the functional user requirements).

4. ACKNOWLEDGMENTS

The speaker is the director of Software Engineering Management Research Laboratory at the Université du Québec à Montréal (Canada). The Laboratory is supported through a partnership with Bell Canada. Additional funding for the Laboratory is provided by the Natural Sciences and Engineering Research Council of Canada.

The author would also like to thank all the members of the COSMIC Group for their thoughtful comments and suggestions in developing version 2,0 of the COSMIC - FFP measurement method.

Up to date information on both COSMIC and COSMIC – FFP can be found on the following web sites:

www.cosmicon.com
www.lrgl.uqam.ca

5 . REFERENCES

ABRAN, A., DESHARNAIS, J M, OLIGNY, S. ST-PIERRE, D., SYMONS, C., *FULL FUNCTION POINTS MEASUREMENT MANUAL VERSION 2,0*, MONTRÉAL, PRE-RELEASE SEPT. 1999.

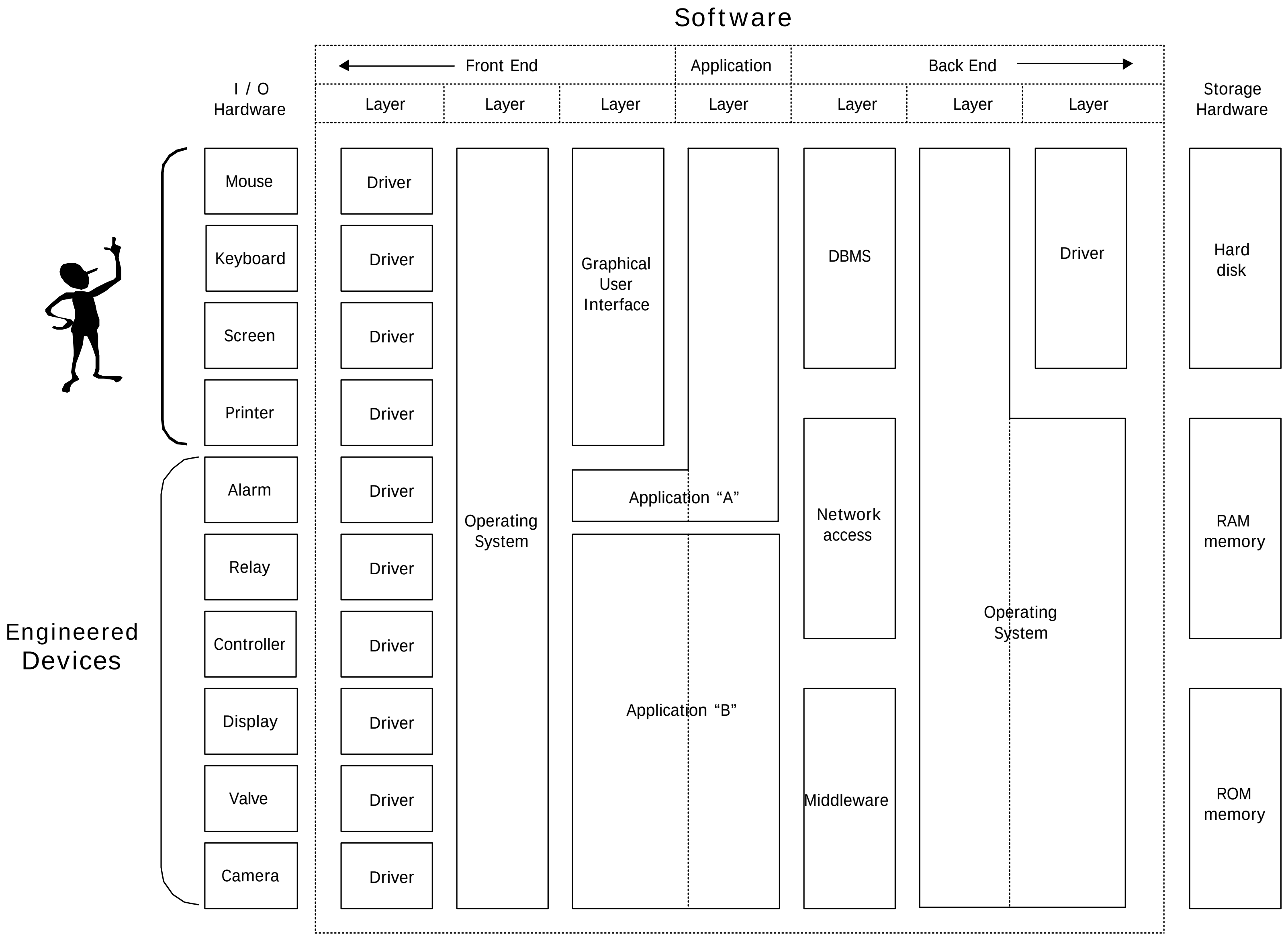
ABRAN, A., M. MAYA et al., *Adapting Function Points to Real-Time Software*, American Programmer, Vol. 10(11) pp. 32-43, 1997.

ISO/IEC 14143-1, *Information Technology – Software Engineering – Functional Size Measurement – Definition and Concepts*, 1998.

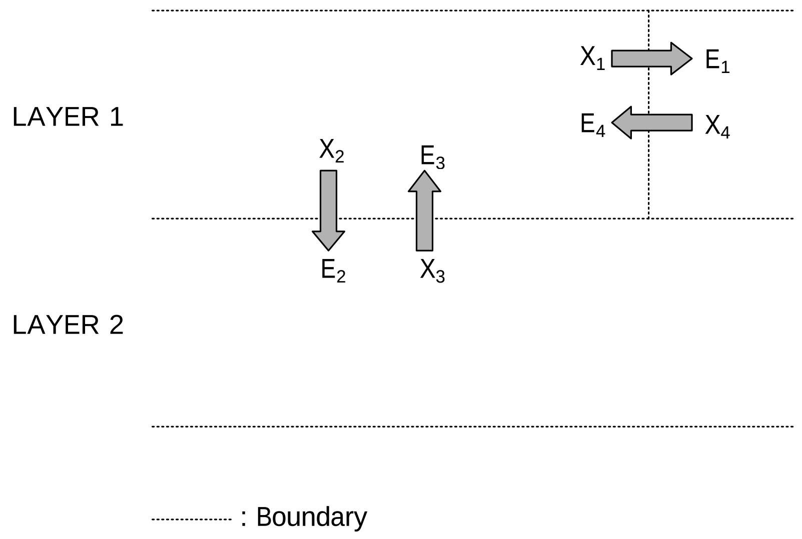
ST-PIERRE, D., MAYA, M. ABRAN, A. & DESHARNAIS, J.-M., *Full Function Points : Function Points Extension for Real-time Software – Counting Practices Manual*, Technical Report no. 1997-04, September 1997.

SYMONS, C., *COSMIC : OVERVIEW OF PRINCIPLES AND STATUS REPORT*, KEYNOTE PRESENTATION, FESMA 99, AMSTERDAM, OCT. 6, 1999

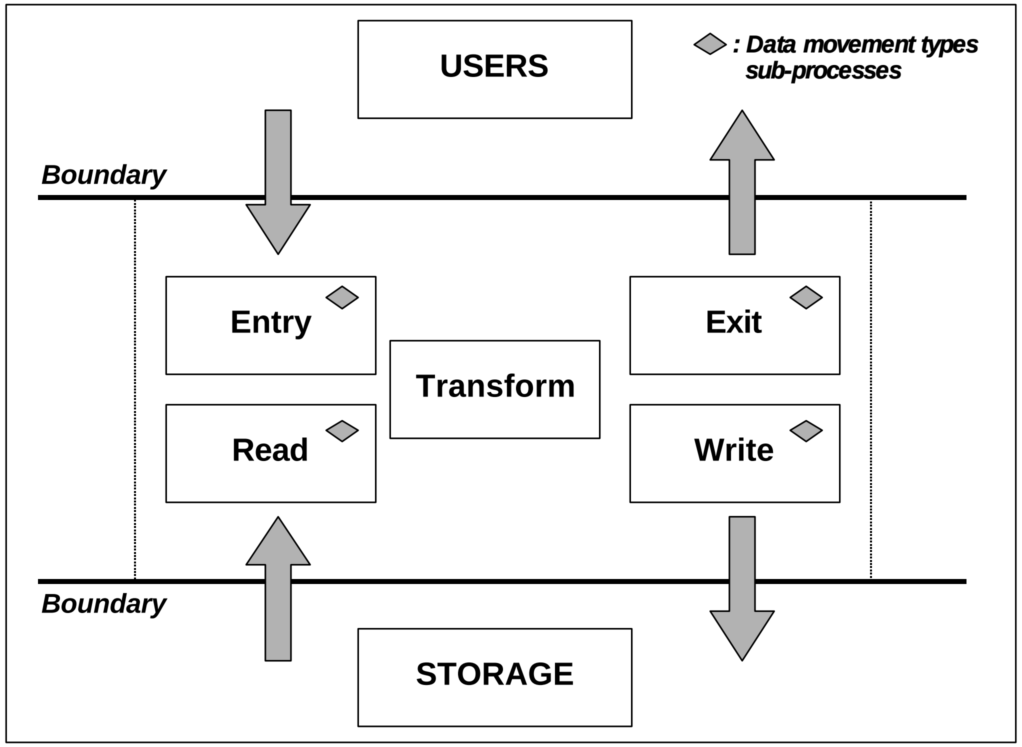
ANNEX A


Software
Front End
Application
Back End
Layer
Layer
Layer
Layer
Layer
Layer
Layer
I / O Hardware
Storage Hardware
Mouse
Keyboard
Screen
Printer
Alarm
Relay
Controller
Display
Valve
Camera
Engineered Devices
Driver
Operating System
Graphical User Interface
Application "A"
Application "B"
DBMS
Network access
Middleware
Operating System
Driver
Hard disk
RAM memory
ROM memory

ANNEX B



ANNEX C


USERS
◇ : Data movement types sub-processes
Boundary
Entry
Exit
Transform
Read
Write
Boundary
STORAGE